U.S.P.S. Express Mail Label No.: El 848969232 US

Date of Deposit: August 22, 2003

ATTORNEY DOCKET NO. 14121US02

FIRMWARE UPDATE NETWORK AND PROCESS EMPLOYING PREPROCESSING TECHNIQUES

CROSS-REFERENCE TO RELATED APPLICATIONS/INCORPORATION BY REFERENCE

[01] This patent application makes reference to, claims priority to and claims benefit from United States Provisional Patent Application Serial No. 60/405,253, entitled "Firmware Update Network And Process Employing Preprocessing Techniques," filed on August 22, 2002, United States Provisional Patent Application Serial No. 60/415,620, entitled "System for Generating Efficient And Compact Update Packages," filed on October 2, 2002, United States Provisional Patent Application Serial No. 60/441,867, entitled "Mobile Handset Update Package Generator That Employs Nodes Technique," filed on January 22, 2003, and United States Provisional Patent Application Serial No. 60/447,977, entitled "Update Package Generator Employing Partial Predictive Mapping Techniques For Generating Update Packages For Mobile Handsets," filed on February 18, 2003.

[02] The complete subject matter of each of the above-referenced United States Patent Applications is hereby incorporated herein by reference, in its entirety. In addition, this application makes reference to United States Provisional Patent Application Serial No. 60/249,606, entitled "System and Method for Updating and Distributing Information", filed November 17, 2000, and International Patent Application Publication No. WO 02/41147 A1, entitled "Systems And Methods For Updating And Distributing Information," publication date March 23, 2002, the complete subject matter of each of which is hereby incorporated herein by reference, in its entirety.

[03] This application is also related to the following co-pending applications, the complete subject matter of each of which is hereby incorporated herein by reference in its entirety:

Ser. No.	Docket No.	Title	Filed	Inventors
	14122US02	System for Generating Efficient and Compact Update Packages	August 21, 2003	Chen Gustafson Barber
	14312US02	Mobile Handset Update Package Generator That Employs Nodes Technique	August 21, 2003	Chen
	14911US02	Update Package Generator Employing Partial Predictive Mapping for Generating Update Packages for Mobile Handsets	August 21, 2003	Lilley

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[04] [Not Applicable]

[MICROFICHE/COPYRIGHT REFERENCE]

[05] [Not Applicable]

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[06] The present invention relates generally to the process of generating and distributing update packages for firmware/software upgrades. The present invention relates, more specifically, to the generation of update packages containing firmware/software version changes, the distribution of such update packages to electronic devices connected to a network, and the subsequent update process of firmware, software and/or content in electronic devices that receive the update packages.

BACKGROUND OF THE ART

[07] Electronic devices, such as mobile phones and personal digital assistants (PDAs), often contain firmware and application software either provided by the manufacturer of the electronic devices, by telecommunication carriers, or by third parties. The firmware and application software often contain software bugs. New versions of the firmware and software are periodically released to fix the bugs or to introduce new features, or both. There is a fundamental problem in providing access to new releases of firmware and software. The electronic devices are often constrained in terms of resources, such as available memory. Attempts to upgrade firmware or software by end-users often result in making the device, or some features of the device inoperable. Specifically, changing firmware in electronic devices requires a great deal of caution as unsuccessful attempts may make the device inoperable. Also, attempts to upgrade firmware and/or software in constrained devices may be hampered by limited user interaction capabilities and slow communication speeds on these devices. In addition, determination of the version of firmware or software that may currently be executed on the electronic devices may not be an easy task, especially if such determination must be made with minimal end-user interaction.

[08] When an electronic device manufacturer/supplier wants to upgrade an electronic device user's executable applications, a binary difference file may be distributed from the supplier to the user. The user may then update the executable image with that difference file. Often, the changes required for the upgrade may be small, however, the binary difference file may be very large, and that may cause problems during the upgrading process.

[09] Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of ordinary skill in the art through comparison of such systems with the present invention as set forth in the remainder of the present application with reference to the drawings.

BRIEF SUMMARY OF THE INVENTION

[10] Aspects of the present invention may be seen in a method for updating firmware/software of an electronic device in a network that comprises the electronic device with an update environment; a distribution environment for transferring data to the electronic device; a communication means for linking the electronic device and the distribution environment; and, a generation environment for generating the data. The update environment of the electronic device may include a download agent for downloading update packages from the distribution environment, and an update agent for adding, deleting or replacing portions of the firmware/software in the electronic device according to the downloaded update package(s). The method of generating an update package for the firmware/software may involve reading the original and new images of the firmware/software; identifying the objects in the images; comparing the objects of the images; and applying bubbles at the appropriate locations in the original image to align objects with the corresponding objects in the new image, until the objects of the two images line up. The original image may be modified by the applied bubbles. The modified original image and the new image may be used to generate an update package with information regarding the differences between the two images. The update package may be transferred to the electronic device via the distribution environment, through the update environment to modify the original image into the new image.

[11] These and other features and advantages of the present invention may be appreciated from a review of the following detailed description of the present invention, along with the accompanying figures in which like reference numerals refer to like parts throughout.

BRIEF DESCRIPTION OF SEVERAL VIEWS OF THE DRAWINGS

[12] Fig. 1 illustrates a block diagram of an exemplary firmware updating network.

[13] Fig. 2 illustrates an example showing the application of a "Worm" bubble, in accordance with the present invention.

[14] Fig. 3 illustrates an example showing the application of another type of a bubble, a "Black" bubble, in accordance with the present invention.

[15] Fig. 4 illustrates an example showing the application of another type of a bubble, a "Mirrored" bubble, in accordance with the present invention.

[16] Fig. 5 illustrates the manner in which bubbles may be applied in a Chaotic image, in accordance with the present invention.

[17] Fig. 6 illustrates utilizing an Annihilation link, in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[18] Fig. 1 illustrates a block diagram of an exemplary firmware updating network 105, in accordance with an embodiment of the present invention. The firmware updating network 105 comprises an electronic device 109 with an update environment 115, a distribution environment 107, a generation environment 111, and a communication means 113 linking the electronic device 109 and the distribution environment 107, in accordance with the present invention. The electronic device 109 may be capable of updating its firmware with the assistance of the update environment 115. The electronic device 109 may download update packages from the distribution environment 107 and apply those update packages to upgrade its firmware/software from one version to another.

[19] The generation environment 111 employs a "Bubbles" technique, explained hereinafter, to create efficient and compact update packages. A "Bubbles" information is generated employing the "Bubbles" technique, the generated "Bubbles" information is subsequently sent to the electronic devices 109 as part of an update package. The distribution environment 107 transfers the update packages to the electronic device 109. Correspondingly, the update environment 115 employs the "Bubbles" technique to preprocess the contents of the electronic device 109 utilizing the update packages received from the distribution environment 107. Thus, the "Bubbles" information and other related information are used in preprocessing activities, explained hereinafter, to prepare the electronic device 109 for an update to a different version of its firmware, software and/or content.

[20] The electronic device 109 with its update environment 115 constitutes a fault tolerant system for updating firmware/software in the electronic device 109. The update package is retrieved from the distribution environment 107.

[21] The update environment 115 facilitates the retrieval of the appropriate update package and its download to the electronic device 109. Upon completion of the download, the electronic device 109 verifies receipt of the right update package, validates the update

package employing CRC checks, etc., and applies the update package to update the version of firmware/software.

[22] In one embodiment, the update environment 115 receives "Bubbles" information and update instructions as part of an update package. The update environment 115 then performs preprocessing steps employing the "Bubbles" information received. Subsequently, the update environment 115 applies the update instructions to upgrade the modified image.

[23] The update instructions may be a set of instructions and data specified employing a specialized set of instructions in a special format. Update instructions may also be a difference file, whereas the generation environment 111 generates such a difference file. Other forms of update instructions are also contemplated.

[24] In one embodiment, the distribution environment 107 may be a carrier network communicatively coupled to a generation environment 111 that is provided by a manufacturer. The electronic device 109 may be communicatively coupled to the carrier network by a communication means 113. In a related embodiment, the carrier network may comprise an update distribution server, an optional generation environment, an optional provisioning system and an optional billing system.

[25] In another embodiment, the distribution environment 107 may be a cable TV network with set-top-boxes with which the electronic device 109 may interact over a communication means 113. The update packages created by the generation environment 111 may be made accessible to the distribution environment 107.

[26] In one embodiment, the update environment 115 in the electronic device 109 may comprise a download agent for receiving a piece of code from the distribution environment 107, and an update agent that may only delete a portion of the firmware in the electronic device 109 upon successful receipt of the piece of code from the distribution environment 107. In a related embodiment, the update environment 115 may constitute a system for receiving a piece of code from the distribution environment 107 and replacing and/or adding only a portion of the firmware of the electronic device 109 upon receipt of the piece of code.

[27] In one embodiment, the update environment 115 may be capable of successfully completing the firmware update process in the electronic device 109 despite interruptions of an earlier attempt to update the firmware of the electronic device 109. Such interruptions may be caused by a loss of connectivity with the distribution environment 107 or a loss of power in the electronic device 109. In this embodiment, the update environment 115 provides for a fault tolerant mode of firmware updates.

[28] In one embodiment, the download of a piece of code by the update environment 115 from the distribution network 107 may be supported employing one of several data transport mechanisms available to the electronic device 109, where the selection of the appropriate data transport mechanism may be based upon several factors. In a related embodiment, the factor for the selection of the appropriate transport mechanism may comprise the available power in the electronic device 109. In another embodiment, the factor for the selection of the appropriate transport mechanism may comprise the speed of any data transport protocol supported and currently available to the electronic device 109.

[29] In one embodiment, after receiving a piece of code from the distribution environment 107, the electronic device 109 verifies whether that the piece of code is appropriate for update of the firmware prior to performing a firmware update. In a related embodiment, verification that the piece of code is appropriate for update of the firmware may comprise verification of location information, CRC checks, hash value computation and comparison, etc.

[30] In addition to being capable of receiving a piece of code and deleting only a portion of the firmware upon successful receipt, the electronic device 109 may be capable of replacing or adding only a portion of the firmware. If the electronic device 109 determines the piece of code to be unsuitable for one or more reasons, the firmware of the electronic device 109 may not be modified and/or added.

[31] The electronic device 109 may perform over-the-air (OTA) firmware updates in a fault-tolerant manner. The distribution environment 107 may make it possible to conduct over-the-air (OTA) updates to firmware, software and/or content on the electronic device 109

by employing one or more of several available communication protocols. The appropriate communication protocol may be selected based on power consumption, bandwidth considerations, etc.

[32] In one embodiment, the generation environment 111 is employed to generate update packages for firmware and/or software that may run on the electronic device 109. The generated update packages are communicated to the distribution environment 107 for testing, storage and eventual distribution to end-users. In a related embodiment, the generated update packages may be electronically transferred (via ftp, http, or other communication protocols) to the distribution environment 107 (such as a carrier network) from the generation environment 111 (such as a manufacturer's update package generation system). In another related embodiment, the generated update packages may be communicated to the distribution environment 107 employing removable media such as a CDROM. In both these related embodiments, the distribution environment 107 stores the generated update packages until they are accessed by the update environment 115 in the electronic device 109.

[33] In general, the electronic device 109 sends a request to the distribution environment 107 to determine the availability of update packages. However, a request may be remotely triggered. For example, a notification from an external system may trigger the electronic device 109 to request update packages.

[34] In one embodiment, the request may comprise a manufacturer identification, a model identification, and a version number. Other parameters may also be included in requests. The distribution environment 107 receives and processes the request, determines the availability of update packages for the electronic device 109, and sends a response indicating whether update packages are available. When the electronic device 109 receives the availability information, it may proceed to request the initiation of the download of the appropriate update package. The download may be over-the-air (OTA) or via wired communication links.

[35] In addition to being capable of updating firmware, the electronic device 109 may also be capable of updating applications, drivers and other software.

PREPROCESSING AND THE "BUBBLES" TECHNIQUE

[36] Preprocessing is a technique that modifies an original version of a firmware image so that it looks more like the new version of the firmware image of an electronic device 109. The modified original version and the new version of the firmware then goes through a generator to produce an update package. This update package is then applied to the firmware in the electronic device 109 to update the firmware in the device from the original to the new version after conducting a preprocessing operation in the electronic device 109. Preprocessing in the present invention is done with the assumption that at least a portion of the objects may appear in the same order in the original and the new images.

[37] In "Branch with Link" Preprocessing, the "Branch with Link" instruction may change according to code shifts in an executable firmware image. "Branch with Link" instructions store relative offsets to other code in the image. If some new information is added between the instruction and its target, then the offset changes accordingly.

[38] When using the "Bubbles" technique, a simple scan of the original image for "Branch with Link" instructions and adjustment of the offset by an amount of shift at that location in the image will usually make the instruction equal to the one in the new image. In cases where constant data that looks like a "Branch with Link" instruction is encountered, running the generator after the preprocessor will recover the proper value.

[39] In "Pointer" Preprocessing, "Pointer" instructions may change according to code shifts in an executable firmware image. "Pointer" instructions load the address of an actual pointer to a register. The value of a pointer at the designated address may be modified if new information is added between the pointer and the location to which it is pointing.

[40] When using the "Bubbles" technique, a simple scan of the original image for "Pointer" instructions and replacement of the offset with the amount of shift at that location in the image will modify the pointer equivalent to the pointer in the new image. In cases where constant data that looks like a "Pointer" instruction is encountered, running the generator after the preprocessing will recover the proper value.

[41] An important part of preprocessing is the "Bubbles" technique. The intent of the "Bubbles" technique is shifting data around in an original image of firmware, inserting spaces and/or removing information to make the data in the original image line up with corresponding data in a new image. The modified original image and the new image may then be sent through a generator to produce an update package. This update package may then be used by an electronic device 109 that has an original image of the firmware, to update the firmware, first by preprocessing (inserting the "Bubbles" into the firmware image) and then by applying the update instructions and data provided in the update package. The "Bubbles" technique of preprocessing assumes that the objects appear in the same order in both the original and the new images.

[42] Auto Bubbles Search (ABS) may be utilized in determining locations of matching objects in the original and the new images. ABS may analyze a constrained range in each image to search for similar patterns in each range. The ranges may be the extent of the entire image, however, better results may be achieved by using regions of the images known to correspond to similar objects. Auto Bubble Search may be used along with an overall bubbles list to remove overlaps in the determined bubbles, which may produce a more optimized bubbles list.

[43] To find patterns in the images, a first step may be to filter the images. For example, for executable code, an effective way to apply a filter may be to disassemble the data and retain only op-codes, which removes address, immediate value, and register information. A next step may be to employ a pattern-matching scheme. Upon iterating through a range in one image, a match may be found in the second image, and a match pattern may emerge as a result. If the length of the match is greater than a certain acceptable value, then it may be considered a match. The aforementioned accepted value may depend on the material being analyzed. For example, setting the acceptable value to thirty-two implies that thirty-two code instruction need to be equivalent for them to be considered matched. Setting the acceptable value to a small number may result in finding a greater number of matches than a case when the acceptable value is set to a larger number. However, setting the acceptable value to a large number may result in overlooking some matches.

[44] The "Bubbles" inserted into the firmware may take different forms. Fig. 2 illustrates an example showing the application of a "Worm" bubble, in accordance with the present invention. The "Worm" bubble may be used if an object in an image grows in size. In an original image 201, an object 209 (b.o) corresponds to an object 211 (b.o) in the new image 203, the new object 211 being larger than the original object 209. The difference in the size between the two objects is equal to a space 207. Adding a space or bubble 213, equivalent in size to space 207, to the old image 201, shifts the objects after object 209, and creates a preprocessed or modified original image 205. As a result, the objects of the modified original image 205 now line up with the objects of the new image 203.

[45] Fig. 3 illustrates an example showing the application of another type of a bubble, a "Black" bubble, in accordance with the present invention. The "Black" bubble may be used if an object in an image shrinks in size. In an original image 301, an object 309 (c.o) corresponds to an object 311 (c.o) in the new image 303, the new object 311 being smaller than the original object 309. The difference in size between the two objects is equal to a space 307. Removing a space or a bubble of data equivalent in size to space 307 from the original image 301, specifically from object 309, shifts the objects of the original image 301, and creates a preprocessed image 305. As a result, the objects of the modified original image 305 now line up with the objects of the new image 303.

[46] In some cases, an object may be added to an image that might have not been in the original image. This may be resolved using a "Worm" bubble, where the added object may be thought of as an increase in the size from the original image, and a bubble may be added to shift the objects of the original image accordingly, resulting in an alignment of the objects of the original and new images.

[47] In other cases, an object may be removed from an image. Hence the new image may not have a certain object that might have been present in the original image. This may be resolved using a "Black" bubble, where the removed object may be thought of as a reduction in the size of the original image, and a bubble or space may be removed from the original image to shift the objects of the original image accordingly, resulting in an alignment of the objects of the original and new images.

[48] Bubbles may also be used in the volatile memory section, such as RAM. These are called "Virtual" bubbles. The "Virtual" bubbles may be "Worm" or "Black" bubbles, but they may not affect the size of an image. "Virtual" bubbles may be used for reference nodes if, for example, the memory allocation for a variable is increased in a new version of a firmware/software. The image of the executable file is not changed, but the reference nodes refer to the variable section that is changed. The "Virtual" bubbles may then be used to make an appropriate adjustment.

[49] Fig. 4 illustrates an example showing the application of another type of a bubble, a "Mirrored" bubble, in accordance with the present invention. "Mirrored" bubbles may be used if a R/W relocation section is used in the image. In the illustration of Fig. 4, the ROM address space 403 contains a R/W section 409 to be relocated to a RAM address space 401. A "Virtual" bubble 411 may be created. Other types of bubbles may also be utilized to achieve the R/W relocation from ROM to RAM. However, there is a mirrored section before the R/W section that may be relocated. This mirrored section should be addressed too. The "Mirrored" bubble 413 reflects the location of the bubble in the RAM section.

[50] Fig. 5 illustrates the manner in which bubbles may be applied in a Chaotic image, in accordance with the present invention. A Chaotic image is one that contains objects that may not be matched between the original image 501 and the new image 503. In order to retain as much information between the two images as possible, a bubble may be applied to align matched locations, such as objects 509 (f.o) and 511 (a.o). Because there are no matched objects between these two objects, a "Black" bubble 507 may be applied to align object 509 in the original and new images, by removing part of object 515 (p.o). As a result, the modified image 505 and the new image 511 have the two common objects 509 and 511 aligned.

[51] Fig. 6 illustrates utilizing an Annihilation link, in accordance with the present invention. An Annihilation link may be used if an object 607 (k.o) appears in one location in an original image 601, but at a different location in the new image 603, while objects surrounding the relocated object remain in the same order in both images. An Annihilation link may be used to avoid destroying the source information by creating a "Worm" bubble

611, and a "Black" bubble 609. The Annihilation link then links the "Worm" bubble 611 with the location of object 607, such that instead of using padding bytes in the original image 601 to shift the objects forward, object 607 will be "inserted" or linked to the appropriate location. Hence, when the "Black" bubble 609 is applied to the original image at the original location of object 607, the objects of the original image 601 will shift backward, and the resulting modified original image 605 will look the same as the new image 603.

[52] The "Bubbles" technique is performed in the generation environment, where the original and new binary firmware images are processed. Based on the object positioning, the original image may be modified. If an object in the new image has grown in size, a "Worm" bubble may be applied at the location of the corresponding object in the original image, and an empty space may be added to the end of it, up to the size difference, shifting all objects after the modified object forward in the original image. If an object in the new image has shrunk in size, a "Black" bubble may be applied at the location of the corresponding object in the original image, and data may be removed at the end of the original object up to the size difference, shifting all objects after the modified object backward in the original image. If an object was added to the new image, an associated bubble may be created in the original image at a location corresponding to the location of the object in the new image. If an object was removed from the original image, then objects that appear after it may be shifted backward accordingly.

[53] The net effect of applying bubbles to an image is to correlate the objects of the original image with those of the new image. The modified original image, or the preprocessing image, then ends up looking very similar to the new image. The generator is then run to build an update package from the modified original image to the new image. The update package along with the "Bubbles" information may then be downloaded, through a distribution environment 107, onto an update environment 105 in an electronic device 109. The electronic device 109 may then use the update package and the "Bubbles" information to update its firmware by preprocessing the original image using the "Bubbles information to create a modified original image. The electronic device 109 then uses the modified original

image with the difference information (for example, a set of instructions and associated data) from the update package to recreate the new image, and as a result update the firmware.

[54] In many systems, a new version of firmware/software may contains elements or objects that may already exist in the present system. Hence, with the "Bubbles" technique only affecting those objects that may have changed, the resulting update packages is often more size-efficient. As a result, the code that need be updated at the electronic device end may be minimal.

[55] The "Bubbles" information may be two integers for each bubble. The first integer may tell where the bubble may be in the original image, and the second integer may be the size of the bubble. As mentioned hereinabove, the size of bubbles may be positive or negative, depending on the alterations in the image. The preprocessor utilizes the identified bubbles to process the original image. Based on the size of a bubble, padding bytes may be added into or information may be removed from the original image. Once the bubbles are applied to the original image, reference nodes within the image need to be re-linked, and a difference file is generated between the modified original image and the new image. This difference file is then delivered to the user side, where the user applies the difference file to get the new image that the supplier provided.

[56] While the present invention has been described with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiment disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.